Java PathFinder: A Tool for Verifying and Validating Software

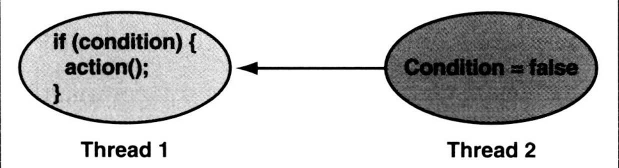
Guillaume Brat, Klaus Havelund, SeungJoon Park, Willem Visser

Software, particularly systems capable of autonomous operation, is fast becoming a major enabling technology at NASA. Unfortunately, the cost savings of autonomous software systems can easily be offset by the risk of in-flight failure of the software. Although rigorous testing of software before deployment can increase the confidence of its correctness, the tendency of in-flight software to be multi-threaded makes it hard to find subtle errors caused by the unforeseen interaction of concurrently executing components. The Java PathFinder project developed two versions of a tool that augments traditional testing techniques in order to find subtle errors in multi-threaded programs.

As can be deduced from the name, the current focus is on finding errors in programs written in Java, but future work will also focus on C and C++ as well as on design notations such as the Unified Modeling Language (UML). Specifically, Java Pathfinder uses a technique called model checking that allows all possible executions of a Java program to be analyzed in order to find errors. Typical errors being targeted include deadlocks and mutual exclusion and assertion violations.

The Java PathFinder project was initiated after a practical experiment in 1997 in which part of the Remote Agent, an artificial-intelligence-based software component of the Deep Space 1 spacecraft, was analyzed as an experiment. The analysis identified five classic multi-threading errors that had not been caught by normal testing. One of these errors is illustrated in the figure which describes a situation in which two threads executing in parallel interact in an unexpected manner. The analysis was done by hand translating part of the Remote Agent code into the language of an existing model-checking tool. This was a time-consuming task. The first version of Java PathFinder, JPF1, finished in August 1999, automated this process by translating from Java to an existing model checker. For the first time, JPF1 demonstrated the feasibility of model-checking Java source code directly without human interaction. JPF1 was applied to two software systems developed at NASA: a satellite file exchange module developed at Goddard Space Center, and a ground control module for the Space Shuttle Launch facility at Kennedy Space Center.

Although demonstrating feasibility, JPF1 had some drawbacks. First, Java is traditionally compiled into byte code (a low-level machine-oriented language) before execution, and libraries often come as byte code rather than as source code. Hence, JPF1 could not handle libraries. Second, since SPIN was used as the model checking engine, and since SPIN is not easily modifiable, it was not possible to experiment with alternative search strategies in order to deal efficiently with large programs. Hence, a new version of Java PathFinder, JPF2, was developed (in Java), which model checks Java byte code directly. Subsequently, advanced testing techniques have been integrated into JPF2, which has been used to find an intricate, but known, error in the real-time operating system DEOS used by Honeywell in business aircraft.



- **Thread 1 tests condition and decides to do action**
- **Thread 2 makes condition false immediately thereafter**
- **Thread 1 now executes action assuming the condition to be true**

Fig. 1. Error pattern found in remote agent.

Point of Contact: K. Havelund
(650) 604-3366
havelund@ptolemy.arc.nasa.gov